AFRL-IF-RS-TR-2004-67
Final Technical Report
March 2004

ORGANICALLY ASSURED AND SURVIVABLE INFORMATION SYSTEMS (OASIS) TECHNOLOGY TRANSITION ASSESSMENT (OTTA)

Wetstone Technologies

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K128, N684, P004

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

AFRL-IF-RS-TR-2004-67 has been reviewed and is approved for publication

APPROVED: /s/

PATRICK M. HURLEY
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** MARCH 2004	**3. REPORT TYPE AND DATES COVERED** Final Sep 01 – Jun 03

4. TITLE AND SUBTITLE ORGANICALLY ASSURED AND SURVIVABLE INFORMATION SYSTEMS (OASIS) TECHNOLOGY TRANSITION ASSESSMENT (OTTA)	**5. FUNDING NUMBERS** C - F30602-01-C-0207 PE - 63760E PR - K128 TA - 00 WU - 01
6. AUTHOR(S) Mike Duren	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Wetstone Technologies 17 Main Street, Suite 237 Cortland New York 13045	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive Arlington Virginia 22203-1714 — AFRL/IFGA 525 Brooks Road Rome New York 13441-4505	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2004-67

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Patrick M. Hurley/IFGA/(315) 330-3624/ Patrick.Hurley@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 Words)*

This effort examined a development beneath the Defense Advanced Research Project Agency's (DARPA's) umbrella program, OASIS (Organically Assured and Survivable Information Systems). The focus of this effort was on the identification of promising technology and assessing the readiness, applicability, and maturity of research and technology derived from or created during the research process. Effective and rapid transition of new technology is essential to the nation's security. With advancements in computing and with States worldwide becoming increasingly technologically sophisticated, we must create an environment that not only fosters cutting edge research, but does so in a way that considers how this research can be deployed most effectively and efficiently. Efficiency must always be a consideration in planning and performing research. It is supposed here that, in some cases, an effective transition environment can improve the quality of research, particularly in the output of research projects, and that early assessment is one means to provide feedback to the research processes at a time when important research decisions are being made or need to be made.

14. SUBJECT TERMS Information Assurance, Technology Transition, Organically Assured, Survivable Information Systems	**15. NUMBER OF PAGES** 56
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

Table of Contents

1 INTRODUCTION 1
1.1 DOCUMENT 1
1.1.1 Purpose 1
1.1.2 Disclaimer 1

2 PROJECT OVERVIEW 2
2.1 BACKGROUND 2
2.2 OBJECTIVES 3
2.3 TASKS AND PRODUCTS 3
2.3.1 Technology Analysis 3
2.3.2 Transition Plans & Designs 4
2.3.3 Integration Experiment 4

3 TECHNOLOGY ANALYSIS AND SELECTION 5
3.1 PROJECT SUMMARY 5
3.2 OASIS PROJECT CLASSIFICATION 13
3.3 SELECTED TECHNOLOGY 17
3.3.1 Autonomix 17
3.3.2 Distributed Intrusion Tolerant Database 18
3.3.3 Aspect-Oriented Assurance 18
3.3.4 PASIS 19
3.3.5 Enterprise Wrappers for Windows 19

4 PROJECT METHODOLOGY 20
4.1 INTEGRATION EXPERIMENT PLAN 20
4.1.1 Integration Target: Seeing Stone 21
4.1.2 Integration Approach 27
4.1.3 Assessment Process 29
4.2 FINAL INTEGRATION STATUS 31

5 FINDINGS 32
5.1 INTEGRATION PROCESS BY SUBJECT 32
5.1.1 Automonix Integration Process 32
5.1.2 Pasis Integration Process 36
5.1.3 Aspects Integration Process 38
5.1.4 ITDB Integration Process 40
5.1.5 Wrappers Integration Process 43
5.2 CONCLUSION 47

6 REFERENCES 50
6.1 PROJECT DOCUMENTS 50
6.2 REFERENCES 50

List of Figures

FIGURE 1 – SEEING STONE OPERATIONAL ENVIRONMENT 22
FIGURE 2 – SEEING STONE FUNCTIONAL ARCHITECTURE 23
FIGURE 3 – INTEGRATION CONFIGURATION 28

List of Tables

TABLE 1 — OASIS PROJECT SUMMARY 5
TABLE 2 — PROJECT CLASSIFICATION: PROJECTS RELATED TO SYSTEM BUILDING 15
TABLE 3 — PROJECT CLASSIFICATION: PROJECTS RELATED TO AIDS 16
TABLE 4 — INTEGRATION STATUS BY SUBJECT 31

1 Introduction

1.1 Document

1.1.1 Purpose

This Program Final Technical Report contains the results and findings for the OASIS Technology Transition Assessment (OTTA) project. It documents the accomplishments of the project and makes recommendations based on the research and development work performed within the project, in its entirety. This report is provided as required by the Statement of Work (SOW) under Air Force Research Laboratory (AFRL) contract F30602-01-C-0207. This report serves as the Contract Data Requirements List (CDRL) CLIN 0002, Item A004 for the said contract.

1.1.2 Disclaimer

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the defense advanced research projects agency or the U.S. Government.

2 Project Overview

2.1 Background

Successful rapid transition of technology to operational environments requires early insertion of promising technologies into military environments, infrastructures, and experiments. In many cases, research that is ongoing can positively impact operational readiness and provide critical input back to the research process. This effort examined a broad range of Information Assurance (IA) technologies, specifically those under development within the Defense Advanced Research Project Agency's (DARPA's) Organically Assured and Survivable Information Systems (OASIS) program.

This effort was focused on the identification of promising technology and assessing the readiness, applicability, and maturity of research and technology derived from or created during the research process. Effective and rapid transition of new technology is essential to the nation's security. With advancements in computing and with countries worldwide becoming increasingly technologically sophisticated, we must create an environment that not only fosters cutting edge research, but does so in a way that considers how this research can be deployed most effectively and efficiently. Efficiency must always be a consideration in planning and performing research. It is supposed here that, in some cases, an effective transition environment can improve the quality of research, particularly in the output of research projects, and that early assessment is one means to provide feedback to the research processes at a time when important research decisions are being made or need to be made.

Assessment can also help determine the relevance and value of a particular technology to the operational theater. It should be cautioned, however, that technology assessment such as was performed on this project is not necessarily an effective means for valuation of a technology or research area. The separation between researchers and implementers is necessary in order to give the research community the creative freedom that enables innovation without significant constraints imposed through operational and deployable system requirements. This is not to suggest that research and operations should not share a common high-level goal. For example, in the OASIS program, the focus of the research is on survivable systems and it is clear how such systems can benefit operational systems.

It is our hope that this project provides and has provided some insightful and useful feedback to individual OASIS projects and to the OASIS program as a whole. In addition, we hope the output from this project is helpful in understanding how the transition process can be improved.

2.2 Objectives

The top-level objective of this project is to assess OASIS technologies for transition readiness. The general process followed for this assessment is as follows:

- Identify OASIS program(s) that fit a model for early transition and adoption
- Develop preliminary designs and recommendations for the integration of the promising technologies into operational environments and exercises
- Facilitate the prototype integration of these results into operational experiments(s).

By defining and executing an operational experiment, selected technologies from OASIS could be thoroughly exercised by integrating them into an existing system. This process enables us to get close to a particular technology and its implementation. As part of this assessment, a well-defined, systematic process was developed in order to provide structure during the assessment. This process also enabled us to report our results in a consistent fashion.

2.3 Tasks and Products

This section contains an overview of the tasks defined for the project.

2.3.1 Technology Analysis

Available OASIS projects are to be reviewed to determine those projects having the highest potential for a successful early transition. Information for the assessment is to be extracted from data generated by the projects themselves, including briefings, demonstrations, reports, and publications, and from interviews with their respective Principal Investigators (PIs), Sponsors, and potential users.

With the aid of AFRL, acquire executable versions of, and documentation for, candidate OASIS products. Meet with the source project teams as necessary to understand how the products can be integrated and/or used with a target system. Run small, preliminary experiments to validate the interfaces and functionality of each candidate. Prepare a report that documents the findings of the product qualification task and identifies any candidates that are not ready for immediate transition.

2.3.2 Transition Plans & Designs

For each high-potential transition candidate, a transition plan is to be developed that designates a transition experiment and the modifications and integration steps necessary to effect the experiment. AFRL, PIs, potential transition partners, and potential users are to be consulted with regard to specific technology selections, experiments, and transition approaches.

Preliminary transition designs are to be prepared for the highest potential candidate(s). Technology(s) will be mapped to a specific operational need, such as an experiment or exercise. The preliminary design is to describe the integration and experiment approach, including consideration of goals, objectives, experiment guidelines, evaluation criteria, experiment definition, and qualified transition partner(s).

2.3.3 Integration Experiment

Build, integrate, and test an integration experiment as planned. Communicate with the OASIS source project teams as necessary. Prepare a report that summarizes the experience of integrating each OASIS product, its behavior, stability, and performance. Demonstrate the integration experiment on two occasions, at AFRL and at an OASIS PI conference.

3 Technology Analysis and Selection

This task has evolved over time as the initial work in the project focused on analysis of the complete OASIS project and all of the research projects therein.

3.1 Project Summary

The following glossary of the current OASIS projects was prepared in conjunction with the initial OASIS project survey using information provided by AFRL. The table served as a quick reference guide when comparing the various projects.

Table 1 — OASIS Project Summary

Project	Contact, URL	Objective	Survivability Focus	Principles & Techniques
A Scalable Intrusion-Tolerant Architecture for Distributed Services (SITAR)	Feiyi Wang, MCNC http://projects.anr.mcnc.org/SITAR/	Develop prototype server cluster based on proxy front ends to redundant COTS servers, voting results, with degree of voting dependent on overall system survivability posture. Also develop system models for prototype architecture to support reasoning about system behavior and system "health"	System design/ composition: servers; also System modeling	Redundancy, graceful degradation, diversity, and dynamism.
Distributed Framework for Perpetually Available and Secure Information Systems (PASIS)	Greg Ganger, CMU; http://pasis.ices.cmu.edu	Apply fragmentation, scattering, redundancy techniques using threshold cryptography to prototype data storage subsystems to assess engineering tradeoffs, usability	System design/ composition: client, server	Redundancy, redundancy management, disperse/ hide sensitive data
Tolerating Intrusions Through Secure System Reconfiguration (Willow)	Alex Wolf, U. CO J. Knight, U. VA P. Devanbu UC Davis http://www.cs.colorado.edu/serl/its/	Develop and demonstrate prototype system showing graceful degradation through reconfiguration as attacks/ failures occur, using Software Dock for software distribution and modeling overall system behavior.	System design/ composition, System modeling/ assurance	Graceful degradation, self monitor and control

Project	Contact, URL	Objective	Survivability Focus	Principles & Techniques
Engineering a Distributed Intrusion Tolerant Database System using COTS Components	Peng Liu, UMBC http://www.research.umbc.edu/~pliu/ItDBMS/index.html	Detect, contain, mask, assess damage, and recover from malicious transactions submitted to COTS relational database.	System design/ composition: application	
Self-Protecting Mobile Agents	Lee Badger, NAI Labs; http://www.nai.com/nai-labs/asp-set/environments/spma.asp http://www.nai.com/ research/ nailabs/ secure-execution/ self-protecting.asp	Develop and prototype agent-based software system supporting computation on potentially hostile platforms, using Aglet infrastructure with heartbeats, periodic re-obfuscation	Programming infrastructure, attack prevention/ system design/ composition: application	Redundancy and redundancy management, self-monitoring, disperse/ obscure sensitive code/ data
Semantic Data Integrity	David Rosenthal, ORA; www.oracorp.com/ http://www.oracorp.com/ projects/ Current/ DataIntegrity.htm	Develop and demonstrate techniques for detecting and repairing damage to the integrity of stored images, includes hierarchical hashing schemes (DSI mark).	System design/ composition: application	Graceful degradation
Cornell On-line Certification Authority (COCA)	Fred Schneider, Cornell University, http://www.cs.cornell.edu/ home/ ldzhou/ coca.htm	On-line Certification Authority		

Project	Contact, URL	Objective	Survivability Focus	Principles & Techniques
New approaches to mobile code: reconciling execution efficiency with provable security	Michael Franz, UCI http://www.ics.uci.edu/~franz/ITS.html	Develop and demonstrate new mobile code transportation schemes that support the deployment of large mobile programs at a much better performance point than current solutions (e.g., Java) and with guaranteed statically verifiable security (i.e., representations that guarantee that any program written in the language will be type-safe).	Programming infrastructure. Error/ Attack prevention. Provide underlying infrastructure for creating, distributing, and executing type-safe programs.	Hardened Core
A Binary Agent Technology for COTS Software Integrity	Dick Schooler, InCert, Anant Agarwal http://www.incert.com/research/cots-int/project.html	Develop and demonstrate technology to instrument pre-existing binaries to detect violations of policy (e.g. out of bounds memory references, including buffer overruns, memory leaks) and report to system monitoring software.	Programming infrastructure. Error/ attack detection and containment. Provide underlying infrastructure for detecting policy violations in running programs	
Scaling Proof-Carrying Code to Production Compilers and Security Policies	Andrew Appel, Princeton, Zhong Shao, Yale U., Ed Felten http://www.cs.princeton.edu/sip/ projects/darpapcc.php3 and http://flint.cs.yale.edu	Develop and demonstrate programming languages and infrastructure to support widespread deployment of type safe mobile code programs together with proofs of advanced security policies that can be checked by the recipient.	Programming infrastructure. Error/ Attack prevention and detection. Provide program development and execution infrastructure.	Hardened core (smaller TCB); abstraction (disperse/ obscure sensitive data); Enforcement via formal specification and verification; self-monitor and control; ACLs & authentication
Sandboxing Mobile Code Execution Environments	Tim Hollebeek, Cigital http://www.rstcorp.com/research/sandboxing.html	Develop software to detect and contain attacks attempting to exploit scripting mechanisms on Windows platforms.	Programming infrastructure: error/ attack detection, containment, reporting.	Access control / intrusion detection

Project	Contact, URL	Objective	Survivability Focus	Principles & Techniques
A Compre-hensive Approach for Intrusion Tolerance Based on Intelligent Compensating Middleware	Amjad Umar, Telcordia http://govt.argreenhouse.com/intrumid/	Develop more a generic approach for robust middleware through application of fragmentation, redundancy, scattering (FRS) techniques to a wide range of COTS middleware technologies including CORBA, Message-Oriented-Middleware (MOM), VoIP, and WAP.	System design/ composition: middleware	Disperse/ obscure sensitive data, deception, graceful degradation, dynamism
Intrusion Tolerance Using Masking, Redundancy, and Dispersion	Janet Lepanto, William Weinstein, Draper Laboratory	Develop and demonstrate system architecture to protect servers against attack, using redundant proxy servers, masking system fingerprint to attackers, and maintaining integrity of COTS backend database.	System design/ composition: servers, application (databases)	Disperse/ obscure sensitive data, deception, diversity, dynamism, self monitor and control, recovery/ restoration
Active Trust Management for Autonomous Adaptive Survivable Systems	Howie Shrobe, MIT http://www.ai.mit.edu/ projects/ its/ index.html	Build Self-monitoring and adaptive systems that detect failures, infer underlying compromises, and steer computations away from compromised or questionable resources.	Programming infrastructure: execution, error/ attack detection, containment, and reporting. Also system design/ composition: application	
Hierarchical Adaptive Control for QoS Intrusion Tolerance (HACQIT)	Jim Just, Teknowledge	Develop prototype survivable COTS-based server cluster for COTS/ GOTS applications behind a firewall and with remote access for critical users via VPN. Cluster employs redundancy, diversity and migration, and decoy servers, internal sensors, adaptive content and separate communication paths for control in effort to meet goal of four hours uptime in the face of red team attack. Not dealing with flooding or other attacks on network infrastructure.	System design/ composition: servers	No single points of failure (redundancy and redundancy management), graceful degradation (reconfiguration, self monitoring and control)

Project	Contact, URL	Objective	Survivability Focus	Principles & Techniques
Intrusion Tolerant Distributed Object Systems	Greg Tally, Network Associates, Inc., -- NAI Labs.	Design and develop prototype intrusion tolerant middleware (CORBA ORB), based on prior fault tolerant CORBA work.	System design/ composition: middleware	1GS: link encryption, 2GS: firewalls; no single points of failure, graceful degradation, diversity, self monitor and control, hardened core (in firewall)
Dependable Intrusion Tolerance	Alfonso Valdes, SRI International http://www.sdl.sri.com/ emerald/	Design and prototype intrusion tolerant server architecture for intrusion detection application; tolerance proxy masks redundant server configuration with degree of operational redundancy/ voting controlled based on detected attack level.	System design/ composition: server; also intrusion detection	
Intrusion Tolerant Server Infrastructure	Dick O'Brien, Secure Computing Corp.	Prototype intrusion tolerant server cluster based on hardened servers and custom hardware at network layer – Policy enforcing NIC card – packet filter controlled from outside host system	System design/ composition: server.	Graceful degradation, no single points of failure, diversity, hardened core
Intrusion Tolerance by Unpredictable Adaptation (ITUA)	Partha Pal, BBN (also U. Illinois, Boeing) http://www.dist-systems.bbn.com/ projects/ ITUA/ index.shtml	Design and develop middleware-based mechanisms to make distributed systems intrusion tolerant using adaptation, redundancy, and uncertainty. and multi-mode redundancy mechanisms that present intrusion-tolerant view of system resources to application.CORBA base, designed to tolerate hybrid faults, combines 1GS and 2GS security mechanisms, brings awareness and control of resources for intrusion tolerance.	System design/ composition: middleware, application objects	No single points of failure, adaptation, uncertainty, dynamism, graceful degradation, incorporate 1GS and 2GS mechanisms

Project	Contact, URL	Objective	Survivability Focus	Principles & Techniques
Randomized Failover Intrusion Tolerant Systems (RFITS)	Ranga Ramanujan, Architecture Technology Corp. (also ORA) http://www.atcorp.com	Develop and document in handbook design patterns for survivable systems resistant to DoS attacks, based on redundancy with failover and recovery process when attacked. Prototype selected survivability design techniques	System design/ composition	no single points of failure, graceful degradation, diversity, dynamism, deception (hiding, obfuscation, dodging)
Applicability of Model Predictive Control (MPC) to Intrusion Tolerance	Pavan Allaghatta or Walt Heimerdinger, Honeywell Labs	Model attack and control of intrusion tolerant system responses using MPC mechanisms.	System modeling/ assurance	self-monitoring and control (esp. closed-loop control); automatic countermeasures (adaptation?) to improve survival probability
Computational Resiliency	Steve Chapin, Syracuse	Intrusion tolerance through replication, migration and recovery of processes when attack is detected. Also formal model using pi-calculus	Programming infrastructure: attack prevention/ detection/ recovery; also system modeling/ assurance	Graceful degradation, Deception, no single points of failure, dynamism
Intrusion Tolerant Software Architecture	Bruno Dutertre / Victoria Stavridou SRI http://www.sdl.sri.com/ dsa/ projects/ itarch/	Develop models of intrusion tolerant system architectures, analyze models using game-theoretic techniques, develop intrusion-tolerant architectures for existing systems (GENOA, SEAS).	System modeling/ assurance	
Survivability Analysis of Networked Systems	Jeannette Wing, Tom Longstaff, CMU	Develop and demonstrate models and methods for analyzing system survivability, incorporating probabilistic behavior and cost functions.	System modeling/ assurance	Dynamism, (attack/ defender/ intruder/ system modeling)

Project	Contact, URL	Objective	Survivability Focus	Principles & Techniques
Dependence Graphs for Information Assurance of Systems	Tim Teitelbaum, Grammatech	Develop tool for exposing control and data dependencies within a software component/ system	Programming infrastructure: error prevention	
Aspect-oriented Security Assurance	Tim Hollebeek, Cigital	Capture security aspects of software development and make available to non-security aware developers	Programming infrastructure: error prevention	Hardened Core at reduced cost
Integrity through Mediated Interfaces	Bob Balzer, Teknowledge http://www.isi.edu/ software-sciences/ integrity-through-mediated-interfaces.html	Develop and demonstrate integrity protection for COTS (MS-Office) application documents on Windows platform, using wrapper technology	Programming infrastructure: error/ attack detection/ containment, also system design/ composition: application	Hardened core (application), disperse/ obscure sensitive data, deception
Enterprise Wrappers for Windows	Bob Balzer, Teknowledge, Mark Feldman, NAI Labs; http://www.nailabs.com; distribution of toolkit and papers at ftp:ftp.tislabs.com/ pub/ wrappers http://www.pgp.com/ research/ nailabs/ secure-execution/ wrappers-overview.asp	Develop infrastructure for distribution and control of wrappers throughout diverse (Unix and Windows) system of systems	Programming infrastructure: development and distribution, execution, error/ attack containment, reporting; System design/ composition: system management	Hardened core (application), disperse/ obscure sensitive data, deception, self-monitor and control

Project	Contact, URL	Objective	Survivability Focus	Principles & Techniques
Autonomix: Component, Network, and System Autonomy	Crispin Cowan, WireX	Develop methods for detecting and preventing damage from commonly exploited software vulnerabilities such as buffer overflows, format bugs, etc.	programming infrastructure: error/ attack prevention and reporting, software execution integrity. Also System design/ composition: clients and servers	
Containment and Integrity for Mobile Code	Fred Schneider, Cornell (Andrew Myers) http://www.cs.cornell.edu/ fbs/ darpaISO.99/ Project.Site.html	Develop concepts and infrastructure to enforce security policies on low-level programs via type safety.	Programming infrastructure: error/ attack prevention and containment.	TCBs, No single points of failure, disperse/ obscure sensitive data, reconfiguration, and static analysis [NOTE: some of these apply to proactive secret sharing work; is this part of this project?]
Intelligent Active Profiling for Detection and Intent Inference of Insider Threat in Information Systems	Joao Cabrera (Scientific Systems) / Lundy Lewis (Aprisma)	Investigate the application of network management systems for the monitoring, detection and response of security violations carried out by insiders.	Programming infrastructure: error/ attack detection, attack response; attack/ fault classification	
A High Security Information System	Joe Johnson, U South Carolina	Assure high availability of Oracle-based operational state emergency management system against both natural and maliciously induced failures; mathematical models of system/ components for evaluation	System design/ composition: application, system management; system modeling/ assurance	1GS, 2GS, no single points of failure, redundancy, redundancy management

Project	Contact, URL	Objective	Survivability Focus	Principles & Techniques
Efficient Code Certification for Open Firmware	Matt Stillerman, Odyssey Research Corp. http://www.oracorp.com/projects/current/EfficientCode.html	Detect potentially malicious firmware (Fcode) programs at boot time by detecting deviations from "type safe" behaviors	Programming infrastructure: error/ attack prevention (low level/ firmware)	hardened core, self-monitoring and control
Novel Applications of Military Science to Intrusion Tolerant Systems	Matt Stillerman, Odyssey Research Corp. http://www.oracorp.com/projects/current/MilitaryScience.html	Identify helpful analogs between conventional military science and cyber warfare, intrusion tolerant systems (e.g., citadels, combined arms warfare, etc.)	System design/ composition	(all? – paper study)
Encoded Program Counter: Self-protection from Buffer Overflow Attacks	Akhilesh Tyagi, Iowa State University	Protect return addresses on stack and function pointers against malicious corruption by encrypting them; attacker cannot alter with predictable results.	Programming infrastructure: error/ attack prevention/ detection	Obscure sensitive data

3.2 OASIS Project Classification

In order to facilitate technology transfer and determine which particular OASIS technologies could be integrated, where, and in what capacity, OASIS projects can be broadly classified according to their functionality. In our approach, we broadly classified OASIS projects at the highest level into two basic categories:

- Projects related to contributions in *System Building*, such as infrastructures, complete systems, or system components
- Projects related to contributions to the *Aids* used for system building and information assurance, such as software tools or methodologies

Furthermore, projects classified as related to 'system building' can be further classified as:

Projects related to *Architectures* and *Systems*, where Architectures refers to theoretical infrastructures, models and frameworks, while Systems refers to implemented systems.

Projects related to design and building of *System Components*.

Each System Building project can be further classified as:

Implementation or Model.

COTS/Middleware or Proprietary

Database-specific application or Not.

Projects classified as related to Aids can be further classified as related to:

Software Checking, such as checking binaries to detect malicious code

System Analysis, such as formal analysis

Design Techniques, such as how to construct correct software systems.

Each Aids project can be further classified as a:

Tool or a Method.

A summary of this classification approach is shown here, and the resulting categorization of the individual projects is shown in Table 2 and Table 3.

OASIS Project Classification Criteria

Level 1	Level 2	Level 3
System Building	• Architectures/Systems • System Components	• Implementation • Model • COTS/Middleware • Proprietary • Database-specific application • Not database specific application
Aids	• Software checking • System analysis • Design techniques	• Tool • Method

Table 2 — Project Classification: Projects Related to System Building

	System Building							
	Level 2 Classification		Level 3 Classification					
Project	Architecture/System	System Components	Implementation	Model	COTS	Proprietary	Database specific	Not database specific
PASIS	✓		✓	✓				✓
SITAR	✓		✓	✓	✓			✓
Willow	✓		✓	✓				✓
Active trust management	✓		✓	✓				✓
Intr. Tol. Server infrastr.	✓		✓			✓		✓
Dependable intr. Tol.	✓		✓					✓
HACQIT	✓		✓		✓			✓
Intr. Tol. Using masking	✓		✓		✓		✓	
Intr. Tol. Distributed Object Systems	✓		✓	✓				✓
Intr. Tol. By Unpredictable Adaptation	✓		✓	✓				✓
High Security Information System	?			✓			✓	
Comprehensive Approach for Intr. Tol.	✓			✓				✓
Eng. Distr. Intr. Tol. Database	✓		✓		✓		✓	
Computational resiliency	?		✓					

Table 3 — Project Classification: Projects Related to Aids

	Aids				
	Level 2 Classification			Level 3 Classification	
Project Name	**Software checking**	**System Analysis**	**Design Techniques**	**Tool**	**Method**
Encoded Program Counter	✓			✓	
Efficient Code Certification for Open Firmware	✓				✓
Enterprise Wrappers	✓				✓
Integrity through Mediated Interfaces		✓		✓	
Autonomix	✓				✓
Self-protecting mobile agents				✓	
New approaches to mobile code				Infrastructure	
Containment and integrity for mobile code				Infrastructure	
Sandboxing mobile code environments	✓			✓	
Scaling proof-carrying code	✓			Infrastructure	
Binary agent technology	✓			✓	
Dependence graphs	✓			✓	
COCA	✓			Infrastructure	
SDI	✓			Infrastructure	

Aspect-oriented sec. assurance			✓		✓
Intr, tol. Software architecture		✓			Modeling Develop architectures
Intelligent active profiling		✓			✓
Applicability of MPC			✓		Modeling
RFITS			✓		✓
Survivability of networked systems		✓			✓
Novel apps. of military science			✓		✓

3.3 Selected Technology

Once the broad categorization of OASIS technologies was made several technologies were honed in on in order to continue the assessment process. The selection process considered maturity, applicability, technology stability, as well as input from AFRL.

The following technologies were selected for the assessment:

- PASIS
- Aspect Oriented Programming
- Autonomix
- ITDB: Intrusion Tolerant Database
- Generic Software Wrappers

The remainder of this section introduces each technology

3.3.1 Autonomix

The "Autonomix" project has sought to improve system survivability by removing vulnerabilities from the operating system and critical applications without requiring special modifications to existing or future applications. Results of this effort have grown into the Immunix family of products, which includes a hardened Linux operating system and various tools to eliminate applications' exposure to common threats. Among the attacks addressed are: buffer overflows, data format corruption, race conditions, and stack overflows.

3.3.2 Distributed Intrusion Tolerant Database

The object of the "Engineering a Distributed Intrusion Tolerant Database System Using COTS Components" project has been to layer intrusion tolerance on top of commercial database systems in order to provide data survivability with no impact to existing database products. The overall effort was divided into six focus areas:

- Transaction-level Intrusion Detection – adapting existing network intrusion detection models to database access
- Intrusion Isolation – "sandboxing" transactions identified as highly suspicious in an isolated environment where they can be executed without harm to operational data
- Intrusion Masking – minimizing the potential damage caused by moderately suspicious transactions at a lower cost than isolation
- Multi-phase Damage Location and Confinement – identifying and confining damage as quickly as possible
- Damage Assessment and Trusted Recovery - assess damage caused from an incident and attempt to repair the system to some known, trusted state.
- Self-stabilization – automatically achieving a known level of data integrity as the environment changes.

3.3.3 Aspect-Oriented Assurance

The goal of the "Aspect-Oriented Security Assurance Solution" project has been to apply the concepts of Aspect-Oriented Programming (AOP) to facilitate the incorporation of security practices and procedures into:

- New code written by security-unaware programmers, and
- Legacy code written with unknown security awareness.

AOP, itself, is a more general programming tool concept in which specific programming concerns, or *aspects*, are defined in an abstract grammar. The scope of an aspect can be narrow, such as substituting one function reference for another, or broad, such as defining a preamble and postamble framework around a sensitive resource reference. Aspects are consumed by an aspect *weaver*, which inspects the original source code and transforms that code in conformance with the aspect. Ignoring specialized terms, the AOP concept is remarkably similar to previous methods such as precompilers and macro-assemblers. In each case, the objective is to expand, harden, or otherwise beneficially interpret original source code, while demanding little or no knowledge of the transformation on the part of the original source programmer.

Products of the Aspect-Oriented Security Assurance Solution project include an aspect grammar for security related concerns, a weaver, and a selection of security-related aspects for application to the *C* programming language.

3.3.4 PASIS

Survivable storage subsystems have been the objective of the "PASIS" project, where survivability is meant to include confidentiality, integrity, and high availability in the face of storage component failures, as well as overt and covert attacks. The design philosophy for PASIS assumes that no service, node, or user can be fully trusted, and that at any time, some subset of the entities within the storage-using community will be compromised. The PASIS approach combines traditional methods of data replication with techniques for distributed secrets to construct experimental storage subsystems that take unique advantage of the secret sharing (m of n) technology for securely distributing and sharing data across multiple nodes. The focus of the project has been to optimize the execution of these systems to achieve acceptable levels of operational performance. Because the commercial server-to-storage interface remains unchanged, PASIS data survivability requires no change to existing applications.

3.3.5 Enterprise Wrappers for Windows

Enterprise Wrappers for Windows, also known as Mediated Connectors, exploits Windows' use of dynamically linked libraries to systematically intervene in routine operations that affect sensitive resources. A framework has been developed that allows the programmer to construct prolog, postlog, or full substitute modules that enforce security policy as required with a minimum knowledge of the wrapper mechanism itself. By focusing on the interface between applications and common Windows systems services, security policies can be developed and enforced without modification of existing applications.

4 Project Methodology

4.1 Integration Experiment Plan

The goal of the OASIS Integration Experiment is to demonstrate the viability of rapidly integrating selected OASIS products into an existing Information Assurance application. Intermediate objectives include determining the applicability of selected products or technologies to an existing system, the ease with which the products are integrated, and, to a lesser extent, the apparent effectiveness of the products with regard to their functional claims. The result may be an assessment of the transition potential for a subclass of OASIS products and technologies.

In order to successfully support the experiment, the target host application must be sufficiently:

- Mature, so that latent host defects do not obscure the new technologies
- Complex, so that several OASIS technologies can be hosted within its architecture
- Straightforward, for the integrator to understand, modify, and demonstrate
- Open, with access to source code and development engineers.

Based on these criteria, as well as the available schedule, WetStone's Network Fuzzy Logic Attack Recognition (NET-FLARE) Intrusion Detection System (IDS) was selected as the target application to incorporate the selected OASIS technologies. The NET-FLARE system was renamed to Seeing Stone and will be described in the following sections. Seeing Stone is a heterogeneous system that includes multiple operating systems, and various communications and data storage mechanisms.

Previously, a list of high-potential OASIS candidates for rapid technology transfer was identified based on a review of available program literature, presentations, and product demonstrations. The objective of the assessment was to select product technologies based on overall maturity and stability, ease of integration with existing software, and functional assimilation with regard to information assurance. Five product technologies were selected as meeting the overall criteria while presenting an acceptable integration-risk profile:

- Autonomix: Component, Network, and System Autonomy
- Distributed Intrusion Tolerant Database System using COTS Components
- An Aspect-Oriented Security Assurance Solution

Perpetually Available and Secure Information Systems (PASIS)

Wrappers for Windows

This Experiment Plan describes the concept, objectives, and approach for integrating these five OASIS product technologies within the existing Seeing Stone architecture.

4.1.1 Integration Target: Seeing Stone

Seeing Stone has grown steadily through three programs with the goal of realizing an Intrusion Detection System (IDS) that simplifies the work of the analyst, by means of computer-aided decision support, and is:

- Flexible in deployment, because few networks are identical or homogeneous
- Scaleable in capacity, because the size range of target networks is large
- Adaptable in detection and correlation, because the threat changes over time
- Field configurable, because analysts have more domain knowledge than engineers
- Easy to use, so that no special skills or training are required for operation.

From its inception, the focus of Seeing Stone's purpose has been policy-based event assessment and that concept has remained at the heart of the system throughout its evolution. Seeing Stone's Policy Editor and Decision Engine together enable the tailored review and disposition of network events in real time. Visualization presents events and event data to the analyst in displays that are tailorable to meet immediate needs. Subsequent programs, NET-FLARE1 and NET-FLARE2, enhanced the power of the Seeing Stone system with increased flexibility, deployability, and decision aides.

4.1.1.1 Seeing Stone Operational Environment

The intended operational environment for Seeing Stone is depicted in Figure 1, which shows Seeing Stone surrounded by the major external components that support or affect its operation.

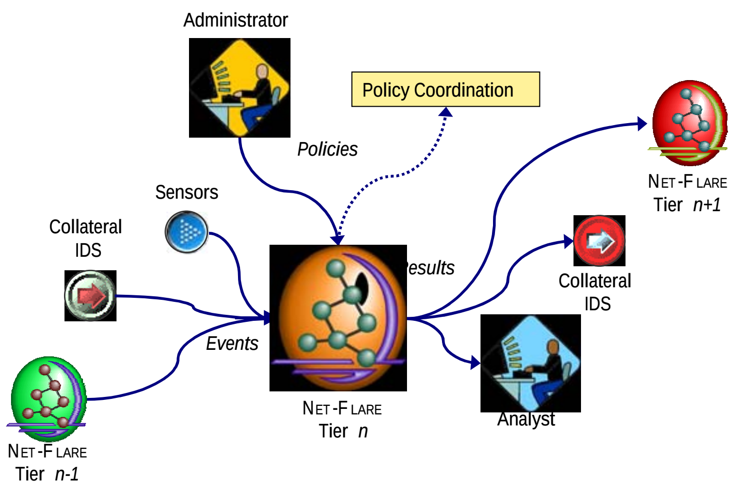


Figure 1 – Seeing Stone Operational Environment

A Seeing Stone administrator controls and tailors Seeing Stone behavior through the creation of Policies using Seeing Stone's graphical Policy Editor. Multiple policies can be created that optimize for specific situations, such as threat conditions or missions, and customized for specific sensors. Policies are loaded and executed on initialization, and subsequently at Administrator command. In support of organizational and distributed deployment, Policies can also be exported and imported, allowing policy coordination among Seeing Stone systems based on doctrine and/or technical performance.

Seeing Stone accepts event inputs from a variety of sources: sensors, collateral intrusion detection systems, and subordinate Seeing Stone systems. Sensors can be either network or host based, and physically remote as well as local, provided that communications between the sensor and Seeing Stone are secure. Other detection and network management systems that produce alarm or event records can be used as event sources by Seeing Stone, in order to layer Seeing Stone capabilities on top of existing legacy systems.

When complex deployments involve multiple organizations, hierarchies of command, or physical distribution, one Seeing Stone system can forward its results to another Seeing Stone that may be serving as a central clearing house or a command-and-control node. Operating under different policies, different Seeing Stone nodes can derive different meanings from the same set of events, depending on their individual missions.

The chaining of Seeing Stone systems is symmetric so that large organizations and hierarchies can be accommodated. Similarly, Seeing Stone results can be forwarded to other detection or management systems in order to support existing, consolidated management and reporting systems. However, the richest access to results is provided to local analysts, who have access to reports and statistics as well as processed event data.

4.1.1.2 Seeing Stone Functional Architecture

Seeing Stone's functional architecture is illustrated in Figure 2.

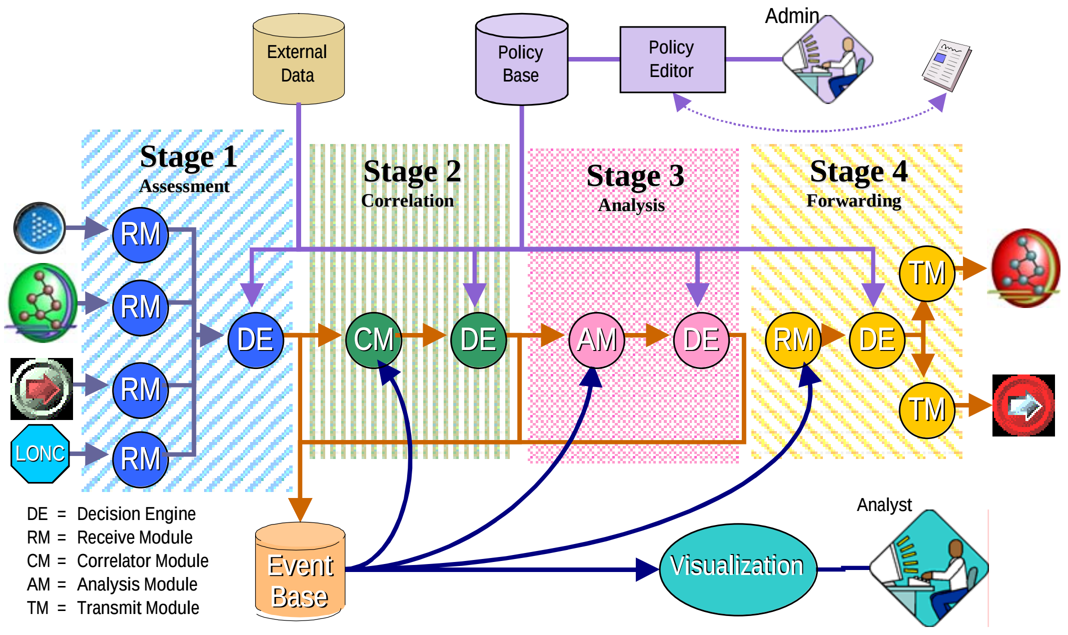


Figure 2 – Seeing Stone Functional Architecture

A Seeing Stone system is a multiprocessing (potentially multiplatform), configuration that consumes events and alarms from sensor systems, analyzes the events singly and in aggregate to identify attacks in real time, and presents status and recommendations to using analysts. The process is organized into a series of four stages, with continuity provided by a central repository containing all events and related decisions. The process is controlled by a central collection of Policies, which guide the disposition of each event and decision.

4.1.1.3 Stage-1, Event Assessment

Event records are received from upstream sensors, collateral systems, or separate Seeing Stone systems for an assessment of significance. Event data, fresh from a sensor, is parsed and normalized by a Receive Module (RM) tailored for that sensor. The resulting raw event record is transferred to the Stage-1 Decision Engine (DE1), which performs a series of steps in accordance with the active Policy for the originating sensor. Raw data is augmented by metadata, then the aggregate is abstracted to create values suitable for fuzzy-logic assessment.

The resulting composite event record containing raw data, metadata, and abstracted data is then evaluated against the active assessment rules for its sensor of origin. When an assessment rule is satisfied, DE1 adds the associated event significance and recommended CoA to the event record. The completed event record is added to the EventBase and forwarded to Stage-2. Designed to dispose of events in near real time, Stage-1 is scalable; many Decision Engines may reside in this stage and assess multiple sensor inputs in parallel.

The LONC, which monitors a Seeing Stone system's own platforms and inter-platform communications, is treated from a dataflow perspective as another symmetric sensor input. Special significance that may be due LONC events is provided by the Policy created for LONC inputs.

4.1.1.4 Stage-2, Event Correlation

Event correlation for the detection of attacks has been separated from analysis processing so that correlations can proceed in parallel, and configurations can scale for anticipated loads. Stage-2 contains one or more Correlation Modules (CMs) and a Decision Engine (DE2). Each CM is structured to detect a particular attack so that the correlator can be focused, configured, and updated as necessary to support specific missions and threat conditions. Based on the input stream of new events from Stage-1 and on reviews of prior events from the EventBase, a CM collects statistics and correlates against attack models using techniques appropriate to the model, which may detect patterns, clusters, trends, or hypotheses.

When an attack is detected, the CM creates a synthetic event, which is then assessed by DE2. Consistent with the basic Seeing Stone approach, significance assessment of Stage-2 synthetic events is controlled by a specific Policy for that activity. DE2 adds metadata, abstracts fuzzy values, and evaluates the entire event record against the rules for attack detection. As in Stage-1, DE2 adds the associated event significance and recommended CoA to the event record, stores the record in the EventBase, and forwards the record to Stage-3.

It is important to note that although events are treated symmetrically throughout Seeing Stone, events can have significantly different semantic values. Synthetic events created in Stage-2 represent not just alarms or observations, but the potential detection of an attack. Similarly, not all Stage-1 events have equal semantic value. In the case of a distributed attack, many observations must be reviewed to detect the pattern; in other cases, the presence of a single, significant observation may indicate attack. It is for this reason that Stage-2 reviews all incoming events after they are assessed by Stage-1.

Stage-2 may contain multiple CMs as necessary to detect anticipated attacks, where some correlations are very specific, while others are more general in nature. It is also anticipated that CMs will respond to different stimuli depending on attack model; for example, some may respond to the arrival of an event record, while others may activate based on elapsed time.

As appropriate for fuzzy-logic assessment, a Confidence Factor (CF) is assigned to each correlation that is carried forward through assessment and into Stage-3. CFs allow evidence to be accumulated and weighed appropriately in subsequent decisions and assessments.

4.1.1.5 Stage-3, Situation/Risk Analysis

The determination of system status with regard to situation and risk is performed in Stage-3 by Analysis Modules (AMs), which are similar in form to CMs, but which operate against different data with different models. Stimulated by the arrival of attack events from Stage-2, by elapsed time, or other factors, AMs identify changes to situation/risk as a function of inputs, current state, and pattern correlation. When a change to situation or risk is detected, the AM generates a synthetic event, which is transferred to the Stage-3 Decision Engine (DE3) for assessment.

Once again, a Policy created for Stage-3 is employed by DE3 to determine the significance of the situation/risk change, following the addition of metadata and abstraction as required. Completed situation/risk events, with significance and CoA, are added to the EventBase.

4.1.1.6 Stage-4, Event Forwarding

The fourth stage supports down-stream reporting of Seeing Stone results to separate Seeing Stone or other collateral systems. For this activity, an RM is employed to monitor the addition of events to the EventBase. New events are transferred to the Stage-4 Decision Engine (DE4), which determines, based on the Policy for each down-stream system, whether or not the event is to be forwarded to that system. Transmit Modules (TMs), in the converse of RM functionality, accept event records, format them for the target down-stream system, and negotiate their transfer.

4.1.1.7 Visualization

Visualization (VIZ) is structured to be responsive to the needs of the analyst to select and report only the events and data of immediate interest. Because the EventBase contains all event records, including both events from external sensors as well as synthetic events generated by attack detection and situation/risk analysis, the analyst has access to critical alerts and correlations using the same mechanisms used against raw events. VIZ is composed of four major subcomponents:

Status – provides an instantaneous overview of situation/risk and attacks in progress

Summary – graphically portrays the statistics and other correlations that result in the current system status

Reports – provide an in-depth review of specific statistics and trends as tailored by the needs of the analyst

Queries – allow the analyst to perform free form, on-line, data mining and correlation in response to spontaneous changes in system conditions, as well as to test new theories and attack models.

4.1.1.8 Policy Administration

Seeing Stone's Policy Editor is used to create all of the policies consumed by the Decision Engines. Multiple policies coexist within a Seeing Stone system. DE1 policies are tailored to each different sensor, and all policies are tailored to different threat conditions and/or missions as appropriate. Regardless of stage, each policy defines three things about the event or synthetic event considered: 1) how raw data, metadata, or statistics are to be abstracted for fuzzy evaluation, 2) rules for classifying the event based on combinations of raw, meta, and fuzzy data, and 3) a recommended CoA for each possible rule result.

No special grammar, predicate language, or knowledge engineering is required to construct policies, and policies may be as detailed as necessary to achieve the desired behavior. Because individual policies can become complex, two built-in tools are available to evaluate policies for completeness and consistency. The completeness check ensures that all possible cases (combinations of data) will result in the firing of a classification rule. The consistency check ensures that only one rule will fire for a given case, and that no rules contain contradictory expressions.

The exchange of policies between Seeing Stone locations is facilitated through import and export functions, which utilize Extended Markup Language (XML) to achieve full expressions of the policies, and easy transport between platforms.

4.1.2 Integration Approach

The overall concept of integrating the subject OASIS product technologies with the existing Seeing Stone IDS application is to apply each OASIS product to a selected Seeing Stone component, so that Seeing Stone sustains no appreciable change its implementation or functional characteristics.

4.1.2.1 Integration Configuration

The five OASIS product technologies are to be applied to Seeing Stone components as shown in Figure 3. In order to avoid unnecessary complexity, the integration target is a minimized configuration of Seeing Stone containing only one stage. However, as the figure indicates, sufficient components remain to serve as targets for each of the subject OASIS subjects. The designated approach for each of the OASIS subjects is outlined in the following sections.

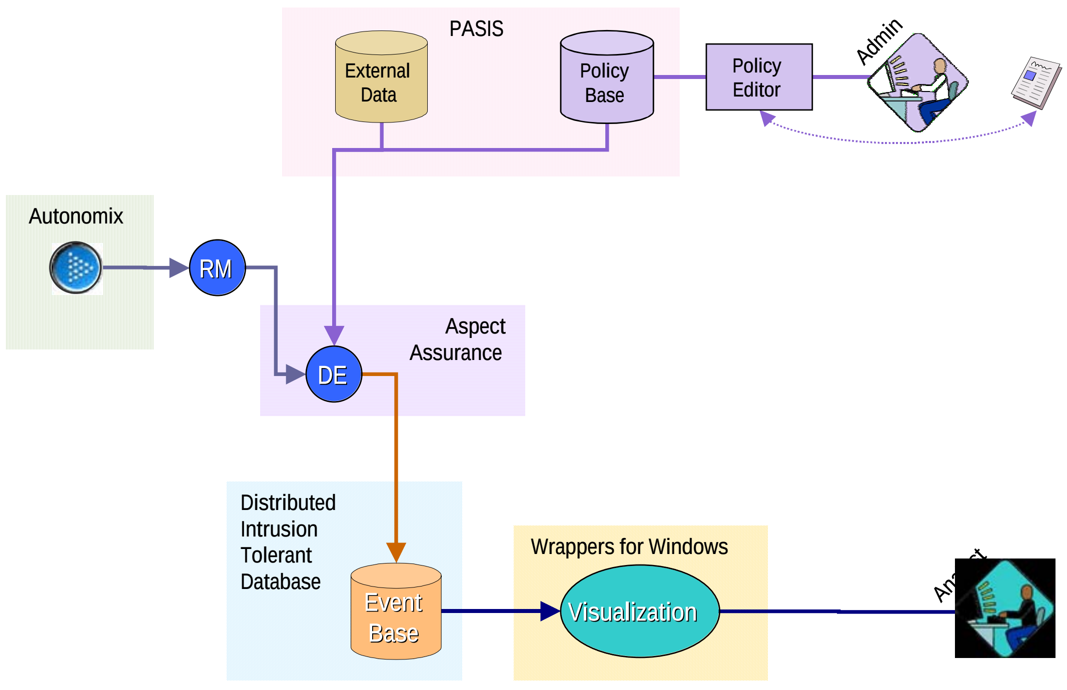


Figure 3 – Integration Configuration

4.1.2.2 Integrating Autonomix

The Seeing Stone system includes a SNORT network-based intrusion detection sensor, the profiles of which have been tuned and optimized for Seeing Stone's use. SNORT itself is an Open Source *C* language component that executes over Linux. Seeing Stone's SNORT component is to be ported to Autonomix's hardened Linux and itself hardened using Autonomix tools as appropriate, such as the SafeStack compiler. No significant changes to SNORT's implementation are anticipated.

4.1.2.3 Integrating PASIS

Seeing Stone Decision Engines obtain policies, initialization data, and external data used in creating event record metadata from conventional files via operating system file services. These files will be ported to a minimal PASIS storage subsystem selected in consultation with the PASIS project team. No changes to the subject files, file access methods, or the Decision Engine implementation are expected.

4.1.2.4 Integrating Aspect Assurance

Based on knowledge of Decision Engine internals, an aspect that addresses a vulnerable or irregular coding construct is to be devised and woven into a broad cross section of the Decision Engine's implementation. No coding changes to the implementation are anticipated to enable weaving, and no functional changes to the resulting component are expected post weaving.

4.1.2.5 Integrating Intrusion Tolerant Database

All Seeing Stone event records and decisions are journalled within the EventBase, which is currently implemented using the Microsoft SQL database product. Distributed Intrusion Tolerant Database (ITDB) components are built to operate over an Oracle database. Therefore, the EventBase schema is to be ported to Oracle, and the Decision Engine and Visualization transactions are to be redirected to the Oracle instance via the ITDB mediation components. Transition of the EventBase from SQL to Oracle is not expected to require implementation changes to either the Decision Engine or Visualization.

4.1.2.6 Integrating Wrappers for Windows

Visualization utilizes data that it does not own extensively. Wrapper(s) are to be developed that enforce communications access policy so that inappropriate data is not subject to compromise through the subversion of the Visualization component. No implementation changes to Visualization are envisioned as the result of wrapper integration.

4.1.3 Assessment Process

A process was required in order to perform the assessment in a structured, well-defined fashion. The focus of the project was not on researching methodologies for assessing technologies, however, we found it interesting to consider such research. It's plausible that a more formal methodology could be developed and/or applied to the assessment process, but consideration would need to be made for the variations in technological approaches, for the fine-grained requirements imposed through deployment, and for the detailed issues that complicate transition. The introduction of a new technology can have significant impact on an operational system. Regardless of the effectiveness and maturity of a specific technology, the impact on a deployed system can be far reaching.

Consideration for operator training, complexity, technological dependencies, and impact on systems must be made.

In this project, a straightforward criterion was defined for the assessment process. The following assessment steps are defined:

Acquisition. Each OASIS subject is to be acquired directly from the source program office along with such documentation as may be available.

Qualification. Each OASIS subject is to be inspected to determine disparities between claims and implementation, as well as release features that may impact the integration process. Detailed integration steps may be based on the findings of subject qualification.

Independence. Each OASIS subject is to be integrated independently, so that the subject can be evaluated without interference from the other subjects.

Static Evaluation. Each OASIS subject is to be evaluated based on the prima fascia effectiveness of the technology with regard to ease of understanding and use, and practicality.

Development Evaluation. Each OASIS subject is to be evaluated based on the actual attempt of integration with its designated Seeing Stone target component, with emphasis on identifying implementation changes that may be required, or other difficulties in rebuilding or reconfiguring the target component.

Dynamic Evaluation. Each OASIS subject is to be evaluated with regard to any adverse functional or operational impact to the target Seeing Stone components.

Security Assessment. It is a second-order objective of this project to assess the security efficacy of each OASIS subject. However, such assessment is optional due to limitations imposed by the effectiveness of the individual integrations as well as test fixtures and/or stimulus that may be required in order to activate the subject under controlled circumstances.

Documentation. Reporting of the complete process from acquisition to transition was on going with details and plans for transition of the OASIS subjects being documented in regular status reports.

4.2 Final Integration Status

During the contract period, integrations proceeded according to plan, with:

Three subjects successfully integrated and dynamically evaluated

Two subjects disqualified for integration.

Details of the integration status are provided in Table 4.

Table 4 — Integration Status by Subject

Subject	Acquisition	Qualification	Static	Development	Dynamic	Security
Autonomix	√	√	√	√	√	√
PASIS	√	√	√	√	√	▨
Aspects	√	√	▨	▨	▨	▨
ITDB	√	√	√	Partial	▨	▨
Wrappers	√	√	√	√	√	√

The two Subjects disqualified for integration were Aspects and ITDB. For Aspects, the methodology and technology were not well developed enough for integration with most operational systems. While there were some questions about compatibilities between the Seeing Stone Decision Engine and Aspects, the primary reason for disqualification was the overall maturity of the Aspects methodology. The concept of the Aspect Oriented methodology is well intended, but the current implementation is weak and incomplete. Also, there are security and architecture concerns in cases where an application is not properly modeled, in particular with multi-programming-language projects.

With ITDB, the approach seems very sound and the sample application provided with the delivery was helpful and demonstrated the ITDB's capabilities and its theoretical approach well. However, the integration of ITDB can be extensive and requires a very in-depth analysis of a subject database and its design in order to determine vulnerabilities and sensitive areas. Our team was able to run the ITDB demo and consider how it might be applied to the Seeing Stone database. We began to modify the Seeing Stone database based on the ITDB model, however, we quickly discovered that a significant amount of analysis was required for proper integration and the work was, therefore, abandoned. It is recommended that more work be performed to assist more general applications in applying the ITDB.

5 Findings

5.1 Integration Process By Subject

This section describes the results of the assessment on a subject-by-subject basis. The assessments are reported in accordance with the assessment process defined in Section 4.1.3.

5.1.1 Automonix Integration Process

Acquisition. Each OASIS subject is to be acquired directly from the source program office along with such documentation as may be available.

The Immunix system is a Linux based operating system. Since this was an operating system different than the target applications operating system, it was necessary to acquire additional hardware to implement this product. Once hardware was identified the ImmunixOS 7.0 was downloaded from the vendor website and installed on an Intel-based computer to be used as the operating environment for the SNORT network sensor integration. Documentation is thorough and updates were easily accessible via Immunix's website. WireX Communications presents this product as a professional and mature solution.

Qualification. Each OASIS subject is to be inspected to determine disparities between claims and implementation, as well as release features that may impact the integration process. Detailed integration steps may be based on the findings of subject qualification.

Research and investigation of the theory behind the security of the Immunix family of product increased the enthusiasm behind this product. Since there are options of how to use the security this system offers, it was decided to try implementing the Snort sensor two ways to verify security enhancements.

Immunix offers protection in its SubDomain kernel for those products that do not allow compilation of source. One installation configuration is to install only the RPM version of SNORT. By doing this it is intended to test the SubDomain kernel that protects a system from vulnerability rot. The classical security solution to vulnerability rot is the notion of least privilege: the technique of granting subjects in a system precisely the capabilities they need to perform their function, and no more. Effective use of least privilege minimizes the potential damage that results when a trusted program is penetrated by minimizing the degree to which the program is trusted.

Stack smashes and format string bugs are popular software vulnerabilities. Immunix 7.0 attempts to limit these attacks by adding FormatGuard and StackGuard to their toolset to assist in compilation of application binaries. To test this security enhancement, the source for SNORT will be compiled and the binary produced will also be tested within the target setup verifying this security enhancement.

Independence. Each OASIS subject is to be integrated independently, so that the subject can be evaluated without interference from the other subjects.

Independently, the implementation was successful. The SNORT sensor binaries were installed and configured likewise; the SNORT source was successfully compiled, installed, and configured. However, some errors had to be overcome.

In the instance of the Source method, the libpcap library and header files were not found on the system. For this specific version of Immunix, the libpcap-0.4 development package was installed but produced errors during the compilation of SNORT. It was necessary to update the library to libpcap-0.6.2-11.7.0.1.i386 rpm before a successful compilation was accomplished. For the RPM method, a similar error occurred because of SNORT's dependence on the libpcap library. Again the updated libpcap library was downloaded from the RedHat site and installed. With this done, the system worked as expected. It is important to note at this point the full benefit of integration with ImmunixOS 7.0 is not achieved since the SNORT and Libpcap-0.6.2-11.7.0.1 binaries are not compiled using the StackGuard and FormatGuard compilers.

As for the implementation of the product, the final product worked successfully giving an alerts file, which could be used by Seeing Stone.

Static Evaluation. Each OASIS subject is to be evaluated based on the prima fascia effectiveness of the technology with regard to ease of understanding and use, and practicality.

Because of the ease with which SNORT was added to the system an additional module was added to the Immunix system called WebMin, which is an administrative interface to Linux. This allowed easier server administration, SNORT administration and also showed yet another example proving the ease with which Immunix's security enhancements could be implemented.

A profile was written for the SubDomain kernel to implement the least privilege security on SNORT as an application. Much work was done to try to counter the security of this system. Snort configuration files were changed to areas of the file system that were not defined within the profile. Not only did the system not allow the action but also SNORT would not start as long as these configurations were in place. However, if the SNORT configuration were defined for any file location allowed in the profile the system would function properly. From an administrative standpoint, this type of security is much easier to manage than User privileges.

Development Evaluation. <u>Each OASIS subject is to be evaluated based on the actual attempt of integration with its designated Seeing Stone target component, with emphasis on identifying implementation changes that may be required, or other difficulties in rebuilding or reconfiguring the target component.</u>

Immunix is a hardened version of Red Hat 7.0. Seeing Stone is hosted on a Windows 2000 Server. Since there is a disconnect between the two operating systems it was necessary to provide a way for the Windows server to read a file, the SNORT Alerts file specifically, on a linux-based system. To solve this problem a Samba server was implemented on the Immunix system. The latest binaries for the Samba server were acquired from the Immunix website to ensure to maintain the security of the system. With Samba in place this provided the ability to serve the SNORT log directory as a windows share. Once the Windows server could access the share, the Alerts file could be configured for Seeing Stone.

This proved to be a simple and sufficient configuration. No modifications were needed on the target system and little effort was needed to integrate the SNORT sensor into the Seeing Stone system.

Dynamic Evaluation. <u>Each OASIS subject is to be evaluated with regard to any adverse functional or operational impact to the target Seeing Stone components.</u>

A network sensor is usually hosted on an external device. Since implementing the SNORT sensor on the Immunix host does not change that, there were no ill effects based on the fact that the system is distributed. Performance tests were run to verify differences between the SNORT sensor run on Red Hat 7.3 and the SNORT sensor run on Immunix

7.0. There did not appear to be any appreciative differences in performance introduced when the binary for SNORT was compiled on the ImmunixOS with StackGuard and FormatGuard. Operationally, the additional 'least privilege' security implemented using SubDomain caused no operational problems as long as all file systems were identified in the profile for SNORT properly.

Security Assessment. <u>It is a second-order objective of this project to assess the security efficacy of each OASIS subject. However, such assessment is optional due to limitations imposed by the effectiveness of the individual integrations as well as test fixtures and/or stimulus that may be required in order to activate the subject under controlled circumstances.</u>

The overall security assessment for Immunix is very positive. The StackGuard and FormatGuard components are easy to understand and use, and have the potential to greatly improve the security of a system, particularly with regard to the damage caused by an attack. As these technologies improve, the Immunix solution can become a very viable Linux solution for all types of applications.

This project did not identify any specific security shortfalls with the Immunix system. There were some concerns and questions, however, that arose with regard to system maintenance and update. The Immunix web site contains detailed information about the baseline Lunix system that was used and what Linux updates are included in the system. This documentation list packages, or Red Hat Package Manager (RPM) packages, that were compiled with StackGuard/Format Guard and included in the Immunix System 7 distribution. According to the Immunix site, many of the Red Hat 7.0 packages (approximately 70) were modified in order to be successfully linked with FormatGuard version of *glibc* which brings up the general question of how an Immunix system is to be maintained and how it can be determined that a given update, or RPM package, can be trusted. This observation can be handled through procedure and based, on the evolution of the Immunix system, it appears they are considering these issues and trying to make the process easier. The latest Immunix version, Immunix Secured OS 7.3, deploys a system utility called *up2date* that, through automation, enables users to more readily see what needs updating on a particular system.

As with any application or operating systems, a well-defined security policy should consider issues about integrity and security. The issue of patch management is particularly difficult and the process of adding security to a baseline OS such as Red Hat Linux can complicate this issue. Users will have to take care to ensure their systems are up to date, not just with the base Red Hat update, but with the Immunix-compiled updates as well.

5.1.2 Pasis Integration Process

Acquisition. Each OASIS subject is to be acquired directly from the source program office along with such documentation as may be available.

Software and documentation acquisition was slow because of unresponsive original requests that stemmed from miscommunications on our behalf. Original requests to the PASIS team were sent to a an incorrect e-mail address. Once contact was made with the PASIS group, the technology was made available almost immediately. This cooperation was very helpful.

Once the documentation was received it was sufficient for theoretical concept, but quite inadequate for implementation. The software was complete and was compiled successfully.

Qualification. Each OASIS subject is to be inspected to determine disparities between claims and implementation, as well as release features that may impact the integration process. Detailed integration steps may be based on the findings of subject qualification.

PASIS requires a considerable amount of analysis to determine the correct implementation. This system is designed to be configurable to the needs of any environment. It has taken much investigation to find the optimal configuration for the Seeing Stone system. However, using the results of the performance tests showed how flexible this system is. Because of the limitation of hardware, storage nodes were limited to 2, so some configurations were not tested.

Problems with the product appear to be caused a lack of maturity. PASIS as tested was very specific in having to match the Linux version the model was created on and unyielding in its configuration setup. Several attempts were made to set PASIS up on other Linux versions causing a lot of investigative time before realizing the software was incompatible with any other version of Linux other than the one the engineering team created their model with.

Independence. Each OASIS subject is to be integrated independently, so that the subject can be evaluated without interference from the other subjects.

The limited and somewhat confusing documentation made setup of the PASIS system difficult. While there is some documentation of the concept of PASIS, actual implementation documentation was inadequate. This caused much trial and error setups to figure out how things worked before determining the best scenario to test under. Once this was accomplished, PASIS was ready for integration with the target system.

Numerous tests were performed to determine the best performance scenario for NET_FLARE. It was discovered during these tests that no matter what the configuration of PASIS, the file system was incapable of handling excessive load. When numerous files were sent to the system, the PASIS system would write only 2 files, one 32 KB file and one 8 KB file and then end with a segmentation fault. There was no debugging or error information provided making the cause of the failure unknown. Since Samba had to be used to make the system accessible to a Windows target, it may have been a factor that contributed to the problems but the objective was to integrate PASIS into our target environment without modifications. This part of the integration left considerable doubt to the usability of the product in a production environment.

Static Evaluation. Each OASIS subject is to be evaluated based on the prima fascia effectiveness of the technology with regard to ease of understanding and use, and practicality.

Static evaluation for PASIS was based on file accessibility. The prior stages addressed the best configuration solution for the target system, but the next step was to find a way to make a LINUX based file system available to a Windows based target. Since PASIS is setup more to the Linux world and with it running NFS in loopback mode on a client, it took some research to find a way to make the file system available to yet another client, a Windows 2000 server. The way this was solved was to use Samba on the PASIS client to distribute the PASIS file system. It is not clear whether or not using this method to distribute the files introduces security risks into the system but with the requirement of least modifications to the target system it was felt this was a viable way to make the file system available.

Once this was accomplished, PASIS functioned as expected and provided a way for Seeing Stone to take advantage of its security enhancements.

Development Evaluation. Each OASIS subject is to be evaluated based on the actual attempt of integration with its designated Seeing Stone target component, with emphasis on identifying implementation changes that may be required, or other difficulties in rebuilding or reconfiguring the target component.

For the development phase, no changes were made to the target system. Since the target system resides in a Windows environment and since the Static phase solved the file accessibility problem this stage had no work required to complete.

Dynamic Evaluation. Each OASIS subject is to be evaluated with regard to any adverse functional or operational impact to the target Seeing Stone components.

While there were considerable hurdles to overcome when configuring the PASIS file system for use with Seeing Stone, the actual effect to the system itself were minimal. There were no changes required to the target other than to point the path locations of policy files and configuration files to the PASIS system. Using PASIS to store these crucial files provided a security factor to the system that would otherwise be at risk on the standard Windows system, especially one with an http port open that is required for Seeing Stone.

5.1.3 Aspects Integration Process

Acquisition. Each OASIS subject is to be acquired directly from the source program office along with such documentation as may be available.

From an acquisition point of view, multiple compilers exist to develop Aspects for different programming languages. This is both good and bad in that the methodology can be applied to many different programming languages but each Aspect language and its associated compiler is specific to the language being targeted.

Qualification. Each OASIS subject is to be inspected to determine disparities between claims and implementation, as well as release features that may impact the integration process. Detailed integration steps may be based on the findings of subject qualification.

The theory of well-defined Aspects like synchronization free a developer up from dealing with adding synchronization software throughout the system and can be applied via an Aspect grammar and compiler. Additionally, primitive security concerns like misuse of buffers or buffer overruns can be developed as Aspects and applied in an automated fashion via an aspect compiler.

However, the approach for developing Aspects is different depending on the language under consideration. For AspectJ the developer writes both the core software and the Aspect software in the same Java like language, and this is then translated into Java. For the Aspect C++ developer the Aspects are developed in an Aspect language and reside in their own files. The native C++ core software is developed in its traditional manor.

All the Aspect grammars appear to support wildcards for developing generic Aspects, although this only works well for very well defined and very limited Aspect functionality. The reason for this is an Aspect has to produce join points in the core software where the Aspect will be called or a pattern match where the Aspect belongs. For even well defined Aspects like synchronization the development of generic Aspects is nearly impossible because the Aspects have to be developed for the data and or methods of objects that require synchronization. While abstracting synchronization into an Aspect is still very useful in that the developer of the core software doesn't need to know how to allow their code to be thread-safe, most complex Aspects require intimate knowledge of the core software that the Aspect is crosscut from, to correctly apply the join points. Finally, Aspect grammars are reasonably complicated and require a fairly large learning curve to become proficient in developing Aspects with them.

Implementing Aspects has its advantages in that debug hooks and logging capabilities can have well defined Aspects and can be applied as needed to legacy and new software. A system whose Aspects are correctly abstracted will sustain its core structure over time as new functionality and or aspects are added because the aspects correctly separate concerns that crosscut the system.

But, the debugging of the system developed using AOP may be more difficult because bugs could be present in both the Aspect grammars and the core software. Also, the bugs may not be obvious in either because they only occur in the merged software hence, debugging of the generated software is required. Debugging generated software is more difficult than developed software due to the fact that readability suffers greatly when dealing with generated software. Finally, adding new functionality to a system developed using AOP may be more difficult in that the new functionality may be required in both the core software and the Aspect grammars. Incorrectly abstracted Aspects will increase this problem by applying the new functionality in many places.

Based on the findings at this stage of Integration Process all assessment efforts for this product were abandoned.

5.1.4 ITDB Integration Process

Acquisition. Each OASIS subject is to be acquired directly from the source program office along with such documentation as may be available.

Acquiring the software and its relative documentation were easily obtained and the point of contact for the deliverables was available and quite responsive to any and all requests. Installation of the software components acquired along with configuration of their demonstration model was adequate but never completely worked. At least one application consistently refused to operate and this problem was never solved.

Qualification. Each OASIS subject is to be inspected to determine disparities between claims and implementation, as well as release features that may impact the integration process. Detailed integration steps may be based on the findings of subject qualification.

The object of "Engineering a Distributed Intrusion Tolerant Database System using COTS Components" has been to layer intrusion tolerance on top of commercial database systems in order to provide data survivability with no impact to existing database products.

All Seeing Stone event records and decisions are journalled with the EventBase, which is currently implemented using the Microsoft SQL database product. Distributed Intrusion Tolerant Database (ITDB) components are built to operate over an Oracle database. Therefore, the EventBase scheme is to be ported to Oracle, and the Decision Engine and Visualization transactions are to be redirected to the Oracle instance via the ITDB mediation components. Transition of the EventBase from SQL to Oracle does require implementation changes. Once the implementation changes are performed, the source code for the ITDB external applications to inject, monitor and repair suspicious activity is needed so modifications can be made to further the experiment and testing.

Finally a determination is needed for the triggers that must be programmed in the new table to key the Intrusion Detection. To do this, much investigation and thought must be put into how a suspicious transaction will be determined.

Independence. Each OASIS subject is to be integrated independently, so that the subject can be evaluated without interference from the other subjects.

Integration with ITDB was more complex than expected. While the implementation theory of intrusion detection on a database is all well proven via their demonstration, much work is needed on any specific application implementation.

Originally, it was theorized that porting the SQL database from Seeing Stone to the Oracle domain would solve the problem of the difference of databases. When that was attempted it turned out to be more complicated. What was then decided was to update Seeing Stone to write directly to Oracle instead of SQL. While this made the integration task easier with respects to database differences and changing the target database may contradict some of the guidelines of this project, it was felt that the change was necessary to further facilitate the evaluation.

Static Evaluation. <u>Each OASIS subject is to be evaluated based on the prima fascia effectiveness of the technology with regard to ease of understanding and use, and practicality.</u>

For this step of the integration process, it was important to understand the concept and usability of the system. To accomplish this it was imperative to get the demonstration model working. The Mediator and Intrusion detector started without any problems, however, the Repair Manager more so than not, would get an "execute statement error" when attempting to send a transaction from the POCI test application. Also no records were inserted into the Repair_det_logs table. The Containment/Uncontainment Manager never worked successfully and after starting the process via the "Start DCMgr" option, the application crashed without an explanation. Finally, the SSM application starts and there is data displayed in the "All Parameters" window, but nothing ever displays in the "SSM Log" window.

Based on these issues several questions were submitted to the development team, these questions are listed below.

1) Is there any more documentation available? The User Manual does not provide much help in how to use the system. The provided manual is more of a demo script than a manual. There are no explanations of what each component is and what exactly it does, nor are there detailed descriptions of the application interfaces.
2) Is there more recent software? The set of files delivered in the tar archives contains the debug versions of the MS C++ development DLLs. Is there a release set of binaries available? Are there newer files?

3) Is there an SQL script (or set of scripts) to build the schemas necessary to run the ITDB system? Since we are integrating ITDB into Seeing Stone, there should be an easier way to get the necessary tables, users, triggers, etc., into the database (i.e., one that is not called ORCINEW).
4) Are there any guidelines on what triggers should be added to my tables in order to invoke the ITDB functionality?
5) Are there any guidelines on how to update the Transaction Patterns file (TranPatt.ctr) in order to get the Mediator to work correctly with Seeing Stone's database?
6) What are the SQL scripts in the Mediator/SQL folder used for? What do I need to place in this folder in order to get the system to work with Net Flare's database?
7) What does the Transaction Type file (Unconfine_Log.txt) do for the DCM application?

Responses from the team were specific and very informative. Not only did they provide answers to the questions, but also provided some updated software, technical design documents and research papers regarding system concepts.

Development Evaluation. Each OASIS subject is to be evaluated based on the actual attempt of integration with its designated Seeing Stone target component, with emphasis on identifying implementation changes that may be required, or other difficulties in rebuilding or reconfiguring the target component.

Once the problems of the previous step were solved, the development evaluation stage began with many obstacles. However, it was expected as the investigation and research required in the static evaluation stage brought to light many areas that would prove to be expensive to solve when integrating the ITDB with Seeing Stone. It was decided to abandon the integration effort because the target application had been altered, additional programming was necessary to produce the triggers for the specific database functions, and insufficient guidelines existed to assist the integrator to fully configure the system.

Having said this, it is important to note that the proof of concept was well done and documented. With more work in the area of configuration, it would be a viable technology. It is not however, ready as a COTS solution due to the problems discussed herein.

5.1.5 Wrappers Integration Process

Acquisition. *Each OASIS subject is to be acquired directly from the source program office along with such documentation as may be available.*

Documentation is bundled with the distribution and contains basic and necessary information for installing and using wrappers. Upon further investigation of the documentation received with the software, some sections of the document were missing and the section of the user manual which describes possible issues with secure wrapped processes does not provide any information about which operations will be considered malicious and should be avoided.

Installation problems ranged from missing dlls to the Control Panel applet never working. This was an error that was overcome by using command line options, but it was a deterrent during installation.

Qualification. *Each OASIS subject is to be inspected to determine disparities between claims and implementation, as well as release features that may impact the integration process. Detailed integration steps may be based on the findings of subject qualification.*

Installation will install documentation, several examples,utility applications for registering/unregistering wrappers and monitoring processes.

NT Wrappers project wizard will be added to Microsoft Visual C++.

Independence. *Each OASIS subject is to be integrated independently, so that the subject can be evaluated without interference from the other subjects.*

A wrapper is implemented by a Windows NT shared library (DLL). Any program construction tool capable of building Windows NT DLLs may be used to build a wrapper's implementation. Authors used (and recommend) WindowsNT SP4 and Microsoft Visual C++ to develop and test wrappers

A Tool that monitors calls made by an application to functions in libraries is provided in the distribution (*Smiley.exe*). *Smiley* will not monitor calls to APIs from static libraries and it will not update itself in response to process creation, termination or dynamic library (un)loading. When *Smiley* starts, the display obtains a snapshot of the active processes on the host. The only circumstance under which a new process will appear in the list is if it is created with *Smiley"s* process creation tool. These limitations imply that we need another tool capable of dynamically monitoring processes. Because of this and because of the decision made to monitor Winsock connections, much investigation was needed to find what exactly was available to the programmer to configure the wrapper for the implementation.

Implementation itself was little more than a skilled programmers task, however, put into a user's hands without these skills makes the integration not only a major task, but could put the system at risk both operationally and securitywise.

Static Evaluation. *Each OASIS subject is to be evaluated based on the prima fascia effectiveness of the technology with regard to ease of understanding and use, and practicality.*

Wrappers support only mediation by process scope. The same wrapper may be active in multiple processes simultaneously if it has been invoked multiple times. Restricting the mediation by thread or trustee can only be accomplished through conditionality in the mediation code itself. Mediating by process scope has a consequence that we have to have a running process in order to install wrappers on it. If a malicious application is started at one point in time on the system, we need a mechanism to wrap it before any call from that application is made. Additional software can be developed that will be notified when the process is started on the system (most likely by WindowsNT system functions or by wrapping a WindowsNT security manager) and then we can try to install a mediator on it.

If a malicious program is wrapped, the wrapper must implement a policy that will restrain its behavior to protect the system. Policies must be written by system and security experts to cover every security critical operation in a system.

It is highly desirable to have source code or protocol specifications for all programs that will run on the system, and since this is not always possible, programs must be reverse engineered or carefully observed during their execution. Observing, no matter how careful, will not guarantee the developer that that application went through all possible states and that he can be certain what DLLs are called by the application. An application might load a DLL at one point during its execution, perform malicious action and then unload the library. In general, starting an malicious program with unknown behavior implies that a wrapper(s) must be installed on that process protecting and monitoring all critical parts of the system, which would introduce overhead.

It is not quite clear which operations are considered malicious by the secure-mode wrapped process (if a secure-mode wrapped process blocks a non-malicious operation it will prevent the process from operating correctly). The blocked operations are ones that are found to be rarely used by non-malicious programs, but it still imposes limits to the developer. Also, it is not possible to mediate functions that accept a variable number of parameters and functions that are returning more than 8 bytes.

When a process with installed wrappers spawns a new process (via one of Window's **CreateProcess** functions), any self-propagating wrappers from the spawning process will be installed in the new process as well. The method by which wrappers are propagated ensures that they are installed in the new process before that process's main thread begins execution. Propagated wrappers will have the same nesting relationships in the new process as they had in the spawning process. No wrapper state is propagated to the new process and removal of a wrapper from a process has no affect on its presence in processes to which, or from which, the wrapper was propagated.

Development Evaluation. *Each OASIS subject is to be evaluated based on the actual attempt of integration with its designated Seeing Stone target component, with emphasis on identifying implementation changes that may be required, or other difficulties in rebuilding or reconfiguring the target component.*

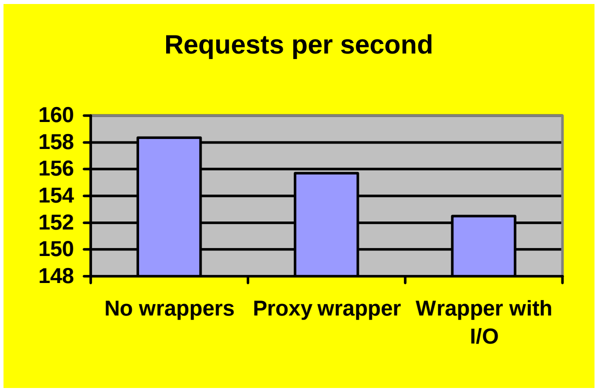
The Seeing Stone target component must have the ability to parse log files created by wrappers that are deployed on the system. We can also reuse existing parsers by creating log files with the same structure as logs that we already have a parser for.

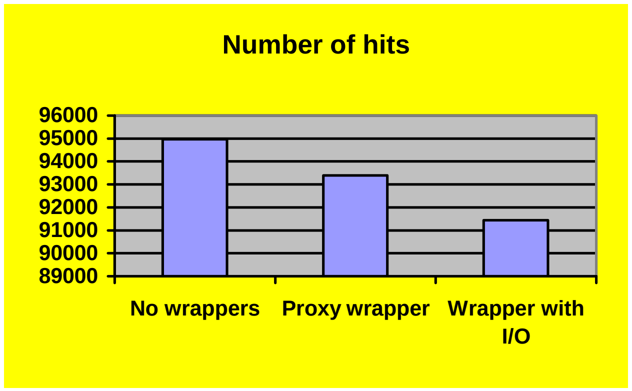
When all security critical operations in a system are identified and we have security policies, it is fairly easy to write a wrapper implementation.

Dynamic Evaluation. *Each OASIS subject is to be evaluated with regard to any adverse functional or operational impact to the target Seeing Stone components.*

The technique used to implement mediators imposes some overhead on each call to a mediated function. This overhead is independent of the number of mediators placed on a function and also independent of the function being mediated, except for a small amount of code that copies parameters and is proportional to the number of bytes of parameter.

Tests were performed using IIS and two different wrappers for winsock DLL send and receive functions. The first wrapper was implemented as a proxy and the second one had a simple file logging capability (for Seeing Stone).





Results show that wrappers introduce 1-2% overhead to the IIS server, which is considered to be minimal.

Security Assessment. *It is a second-order objective of this project to assess the security efficacy of each OASIS subject. However, such assessment is optional due to limitations imposed by the effectiveness of the individual integrations as well as test fixtures and/or stimulus that may be required in order to activate the subject under controlled circumstances.*

Wrappers can protect an underlying operating system from malicious programs but it is assumed that the underlying operating system is functional and that there are no processes with enough privileges to turn the wrappers off or uninstall them. To prevent malicious processes to perform certain operations that would circumvent or nullify the effect of a wrapper every wrapped process can be wrapped in secure mode. A secure-mode wrapped process expends a small amount of resources attempting to detect and block these operations. Since it is not generally possible to determine with absolute certainty whether an operation is malicious – or more generally, part of a malicious sequence of operations – it is possible that a secure-mode wrapped process will block a non-malicious operation, preventing the process from operating correctly.

Wrapper implementation must not introduce any additional security risks and it needs to be designed and coded carefully. Any errors that will result in wrapper termination will also terminate the whole application that has a wrapper registered to it. If that happens, an additional mechanism must be in place to perform integrity checks on the application. It must be applied to both the wrapper and system data files.

This technology can be used for a wide variety of security applications. Wrappers are already developed for safe email attachments, safe web browser activity, safe office operations, executable corruption detector, protected path, local/remote process tracker, no interprocess meddling…and much more. It is obvious that this is a valuable security enhancement when integrated with considerable skill.

5.2 Conclusion

In these parting remarks, we would first like to thank all of the OASIS researchers for working with us in our assessment; their cooperation enabled us to perform this work. While our results tend to focus on difficulties encountered in using and deploying the selected technologies, this is not to say that a given technology or a given research project is not valuable. The relative maturity of the OASIS technology implementations varies significantly from project to project and that maturity level greatly affects the degree to which a technology can be deployed.

In general, the maturity levels of the technology from most the Subjects were low. This is to be expected as researchers are trying to prove their methods or innovative approaches. Researchers innovating to create new, more advanced technologies are focusing on new methodologies and on sound research principles that will help ensure that a given approach is viable on its own. However, for effective and rapid deployment for newly researched technologies, consideration must be made for real-life environments and, in the case of survivable systems, the operational environment must be considered during the research process. This presents a problem in that researchers are typically not skilled implementers and do not necessarily possess the same experience as an expert in the field and their goals are not necessarily the same.

For deployment, system designers must often consider the more broad impact that a particular piece of technology has on a system, such as training, complexity, or interoperability. Also, quality standards such as SEI-CMM or ISO/IEC 9001 for software and system development are often important considerations for government and commercial software development efforts and system integration jobs. In military applications, the DITSCAP process is an integral part of operational system security and focuses on the technology being deployed, the target operational environment requirements, and the defined processes that go along with a particular system. These processes are all considerations that are made at some point in the deployment process, whether it is during design and development, system planning and deployment, or system operation. It is argued here that in most circumstances researchers cannot be restrained by such considerations as the CMM or DITSCAP. These processes are complex and their influence on the research process would be negative.

To overcome this problem, it is suggested that researchers team more closely with developers and integrators during the research process. Through teaming or through separately funded efforts, skilled implementers can work directly with researchers to develop quality implementations that represent the work of the researcher and that consider deployment issues. Within this operational experiment for the OASIS program, we have had the opportunity to experience deficiencies in the technology resulting from research. In all cases, the research that was studied here was well-intended and well thought out. Unfortunately, most of the technological components from the projects were lacking in one sense or another.

There was however, one project that set the standard for maturity and ease of deployment.   The Autonomix project developed technology that was mature, well documented, and easy to use. Its effectiveness and success in this assessment is not just based on the maturity of its implementation, it is also based on the results of our static evaluation. The Autonomix system is autonomous and its deployment does not introduce significant complexity into a system or network of systems. In addition, the system does not introduce significant interdependencies on other systems and can be deployed by swapping out base Linux systems. Note that it’s clear the Autonomix system has commercial potential and is being licensed commercially. It’s not clear to what degree the maturity of the technology has been, or will be, instrumental in the commercial success of Autonomix or if the commercial applicability of the technology and the commercial ambitions of the Autonomix author forced the technology to reach a higher level of maturity, however, one might postulate that there is a relationship between these factors.

In conclusion, we have determined that the largest inhibitor for effective and efficient assessment is in the maturity of the OASIS technologies. This translates directly to the deployment problem, specifically how closely the performed operational experiment mimics actual deployment. Logically, if considerable time is spent achieving operational status in a mock environment, the same will hold true during a real transition.

During this project, OASIS researchers were quick to point out that their implementations might not be sufficiently mature. Although implementation maturity is not necessarily high on the priority list for researchers, certainly placing more emphasis on implementation will serve to facilitate transition into the operational environments demanding new and innovative approaches.

Clearly, there is a need for more early-on consideration of transition issues and technology maturity. The process must definitely improve in order to allow operational experiments such as the one performed under this contract, to effectively provide more insight and ultimately improve the *research – transition – operation* pathway.

6 References

6.1 Project Documents

Note: The following referenced documents were previously delivered under this effort.

- Program Progress Reports. Recurring monthly; contractor format, electronic delivery.
- Presentation Material. As required; contractor format.
- Technical Information Report: *Selection Criteria / Prioritization.* Initial, 18 January 2002; Final, 18 September 2002.
- Scientific & Technical Report: *Experiment Summary Report.* 18 September 2002.
- COTS Manuals and Associated Data. As required; commercial format(s).

6.2 References

[COCA] Cornell On-line Certification Authority (COCA) home page, http://www.cs.cornell.edu/home/ldzhou/coca.htm. Last updated on April 25, 2001.

[DITP] P. Liu, UMBC Lab for Information and System Security, 'Engineering a Distributed Intrusion Tolerant Database System using COTS Components," http;//www.research.umbc.edu/~pliu/ItDBMS/home.html.

[PASIS] CMU PASIS home page, http://www.ices.cmu.edu/pasis/. Last updated on July 17, 2001.

[SDI] Odyssey Research Associates, Inc., "Semantic Data Integrity." http://www.oracorp.com/projects/current/dataIntegrity.html. Last updated January 2, 2001.

[SITAR] R. Wang, MCNC, "SITAR – a Scalable Intrusion Tolerant Architecture." http://projects.anr.mcnc.org/SITAR.

[SPMA] Network Associates Technology, Inc., "Secure Execution Environments: Self-Protecting Mobile Agents," http://www.nai.com/research/nailabs/secure-execution/self-protecting.asp. Last updated 2001.

[Willow] Software Engineering Research laboratory, University of Colorado, ‘Tolerating Intrusions Through Secure System Reconfiguration.” http://www.cs.colorado.edu/serl/its. Last updated 2000.

[OASIS] OASIS Site, ‘OASIS Project Summaries’, http://www.tolerantsystems.org/ProjectSummaries/Project_Summaries.html